

DERGMs: Degeneracy-restricted exponential random graph models

Vishesh Karwa and Sonja Petrović and Denis Bajić*

Abstract: We propose a new exponential family of models for random graphs. Starting from the standard exponential random graph model (ERGM) framework, we propose an extension that addresses some of the well-known issues with ERGMs. Specifically, we solve the problem of computational intractability and ‘degenerate’ model behavior by an interpretable support restriction.

1. Introduction

Exponential family random graph models, also known as ERGMs for short, are known to be a theoretically flexible class for modeling and estimating network features. There is a growing literature in applications such as [Snijders et al. \(2006\)](#), [Saul and Filkov \(2007\)](#) and [Goodreau, Kitts and Morris \(2009\)](#), but also a growing set of contributions on concerns on model complexity and degenerate behavior. This is not surprising, as such flexibility inevitably comes at a price. Among the many contributions, we single out recent work by [Yin, Rinaldo and Fadnavis \(To appear\)](#), [Chatterjee and Diaconis \(2013\)](#), [Bannister, Devanny and Eppstein \(2014\)](#), where various issues have been pointed out and addressed theoretically. While ERGMs may, as some like to phrase it, ‘behave badly’, this literature also suggests that if we understand this bad behavior, we can still work with this model family - a desirable outcome as the family is so flexible and broadly encompassing. Our work contributes to this understanding and proposes a general modification of ERGMs that both solves the intractability and degenerate behavior issues and is interpretable in applications.

Degenerate behavior of ERGMs, as explained in [Handcock et al. \(2003\)](#) and, more recently, in [Rinaldo, Fienberg and Zhou \(2009\)](#), stems from the model placing most of the probability mass on a small region of the support. Moreover, the subset of parameters where the bad behavior does *not* happen is very small. This property is then implicated in other problems such as estimation, in particular, non-convergence of MCMC-MLE estimates. Estimation is done by approximating the log likelihood using importance sampling from the model with a fixed parameter θ_0 , which is usually done by using an MCMC sampler. To obtain an accurate approximation of the log likelihood, the MCMC sampler must generate samples from the region where the mass is concentrated. Since the mass is tightly concentrated on a small region, the MCMC sampler must start with a parameter very close to MLE, otherwise estimation fails. We resolve both of these issues by using a combination of support restriction and a Monte Carlo sampler. The support restriction allows the mass to be more uniformly distributed, thus reducing the bad behavior. A uniform sampler can then be used to approximate the likelihood without the need of a starting point close to the MLE.

To decide how to restrict support, we consider one particular notion of graph sparsity, and we do so for two reasons. First, as it has been noted in much of the network literature, many real-world networks are sparse in some sense. Intuitively, sparsity means that the nodes in the network have a relatively small number of neighbors. While there are many different notions of sparsity, we consider the following: a network is said to be sparse if it has bounded maximum degree, since bounded maximum degree means that all vertices have few neighbors. More technically, a graph is said to be *k-degenerate* – where this graph-theoretic degeneracy must not be confused with statistical degeneracy! – if every induced subgraph has a vertex of degree at most k ([Seidman \(1983\)](#)). Graph degeneracy has other characterizations, for instance, a *k-degenerate* graph admits an ordering of its vertices v_1, \dots, v_n such that vertex v_i has at most k neighbors after it in the ordering; thus a bounded-degeneracy graph means there exists a vertex with few neighbors. Hence, bounding the degeneracy

*Equal contribution authorship. V.K. is with Department of Statistics, Harvard University; S.P. is with Department of Applied Mathematics, Illinois Institute of Technology; D.B. is a Masters student in the Department of Computer Science, Illinois Institute of Technology. This work is supported by U.S. Air Force Office of Scientific Research Grant #FA9550-14-1-0141 to IIT. A small subset of the simulations for this work were completed on IIT’s Karlin cluster.

of a graph is a *weaker constraint* than bounding the overall node degree in the graph, and it is also weaker than bounding the so-called *h-index*, which means that most nodes have few neighbors. The second reason we consider this support restriction is that restricting to bounded-degeneracy graphs makes many algorithms simpler: for example, all the maximal cliques can be enumerated in polynomial time in the case of bounded degeneracy, while in general the problem is NP-hard. As we will show, while ERGMs are NP-hard to fit, ERGMs restricted to degenerate graphs come equipped with *polynomial time* sampling algorithms instead.

To this end, we introduce a new class of models that we call *degeneracy-restricted ERGMs* or DERGMs.

Let \mathcal{G}_n be the set of all graphs on n nodes. Recall that the ERGM with sufficient statistics vector $t = (t_1, \dots, t_d)$ defined on the parameter space $\Theta \subset \mathbb{R}^d$ places the following probability on any $g \in \mathcal{G}_n$:

$$P_{\text{ERGM}}(G = g) = \frac{\exp\{\theta^T \cdot t(g)\}}{c(\theta)}, \quad (1)$$

where $\theta = (\theta_1, \dots, \theta_d)$ are the canonical parameters and $c(\theta)$ is the normalizing constant $c(\theta) = \sum_{g \in \mathcal{G}_n} \exp\{\theta^T \cdot t(g)\}$. In the corresponding DERGM, we simply restrict the support of the model from \mathcal{G}_n to the set of all graphs on n nodes whose degeneracy is at most k .

Definition 1 (DERGM). *Denote by $\mathcal{G}_{n,k}$ the set of all graphs on n nodes whose degeneracy is at most k . Choose a vector of graph statistics $t = (t_1, \dots, t_d)$. The degeneracy-restricted exponential random graph model, or DERGM for short, with sufficient statistics vector t places the following probability on a graph on n nodes:*

$$P_{\text{DERGM}}(G = g) = \begin{cases} \exp\{\theta^T \cdot t(g)\} \cdot c_k(\theta)^{-1}, & \text{if } g \in \mathcal{G}_{n,k} \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where $c_k(\theta)$ is the modified normalizing constant

$$c_k(\theta) = \sum_{g \in \mathcal{G}_{n,k}} \exp\{\theta^T \cdot t(g)\}.$$

Note that setting $n = k - 1$ reduces the DERGM to the usual ERGM, thus this is a more general model class.

In Section 2 we discuss the general estimation problem in DERGMs and address various aspects of the problem. Specifically, we discuss the choice of k , why DERGMs do not suffer from the same estimation issues that arise in standard ERGMs, model degeneracy issues and how they disappear for smaller values of k , and the advantage of Monte Carlo estimation over MCMC. As shown in Handcock et al. (2003), the difficulty in approximating the likelihood using MCMC samples from \mathbb{P}_{θ_0} is inherently tied to the ‘bad behavior’ of the ERGM. By restricting the model support, we eliminate some of this bad behavior; by using an uniform Monte Carlo sampler, we eliminate the disadvantages of the MCMC sampler. We discuss the bad behavior of the model and illustrate that the support restriction is a natural solution to this problem. In summary, we argue that small values of k lead to better behaved models, thereby making DERGMs of interest when standard ERGMs do not behave well. In addition, many real-world networks have low observed degeneracy, making the DERGMs no less reasonable than any usual ERGM, yet better behaved. For supporting evidence of low-degeneracy network data, see (Karwa et al., Section 3.1).

In Section 3 we provide simulation results that support the theoretical claims about degeneracy-restricted ERGMs. There, we focus on a specific model as a running example, namely, one with a two-dimensional parameter space whose sufficient statistics are the number of edges and number of triangles in the graph. These are well-studied sufficient statistics that arise naturally when considering Markov dependence, see for example Frank and Strauss (1986). In addition, the edge-triangle ERGM is a running example in Rinaldo, Fienberg and Zhou (2009), where the authors discuss model degeneracy, and as such is a natural example to compare ERGM behavior to DERGMs.

Section 4 derives uniform samplers of the sample space $\mathcal{G}_{n,k}$ and further discusses the algorithmic developments. Finally, Section 5 provides a straightforward Metropolis-Hastings algorithm to sample from the model.

The `R` and `Python` code used to run the simulations in Section 3, along with implementations of the main algorithms from Section 4, is available on [GitHub](#) under [Bajić \(2016\)](#). Some practical computational considerations are discussed in Appendix A (Section 7).

2. Maximum Likelihood Estimation of DERGMs

Consider the problem of estimating the parameters of a DERGM given by Equation (2) from a single observed graph g_{obs} on n nodes. (In the unlikely event that the sample size is larger, one may average.) Suppose that g_{obs} is of degeneracy k_{obs} . To specify a DERGM, we need to specify the parameter vector θ and the degeneracy parameter k . The problem of simultaneously estimating θ and k from g_{obs} seems quite difficult, similar to the problem of simultaneously estimating the parameters of a mixture model and the number of components. The choice of k is thus akin to the problem of model selection, as different values of k describe different models. We will discuss the choice of k later, but for now note that k needs to be at least k_{obs} , otherwise, the model places 0 probability on the observed graph. On the other hand, setting $k = n - 1$ gives us the usual ERGM. Setting $k = k_{obs}$ seems to be a reasonable choice. From now on, assume k is fixed and equal to k_{obs} . For a fixed k , one can write the log-likelihood function of a DERGM in the following form:

$$l(\theta; g_{obs}) = -\log \left(\sum_{g \in \mathcal{G}_{n,k}} \exp(\theta^T \Delta(g; g_{obs})) \right), \quad (3)$$

where $\Delta(g; g_{obs}) = t(g) - t(g_{obs})$. We will also use $\Delta(g)$ to denote $\Delta(g; g_{obs})$ when it is clear that g_{obs} is fixed. The maximum likelihood estimate of θ is

$$\hat{\theta} = \arg \max l(\theta, g_{obs}).$$

As is the case with ERGMs, directly maximizing Equation (3) to obtain $\hat{\theta}$ is intractable, and we need to resort to *approximate likelihood* methods; see, for example, [Geyer and Thompson \(1992\)](#). The key idea is to approximate the log likelihood function using importance sampling, which is then maximized. For the usual ERGMs, the importance samples are generated from the model at a fixed parameter θ_0 using MCMC methods, see [Hunter and Handcock \(2006\)](#).

Instead of sampling from the model using an MCMC sampler, we use a Monte Carlo sampler to sample from a uniform distribution and estimate the likelihood of the DERGM. The combination of support restriction by using $k < n - 1$ and the use of a Monte Carlo sample from a uniform distribution allows us to resolve some of the issues of degeneracy in specification and estimation of the usual ERGM; further details on this are in the discussion below. To this end, consider the uniform distribution $\mathcal{U}_{n,k}$ on the set $\mathcal{G}_{n,k}$. Let G be a random graph on n nodes drawn from $\mathcal{U}_{n,k}$. The log-likelihood from Equation (3) can be written as:

$$l(\theta; g_{obs}) = -\log(|\mathcal{G}_{n,k}|) - \log \mathbb{E} [\exp(\theta^T \Delta(G; g_{obs}))], \quad (4)$$

where $\Delta(G; g_{obs}) = t(G) - t(g_{obs})$ and the expectation is over $\mathcal{U}_{n,k}$. If G_1, \dots, G_B are iid samples from $\mathcal{U}_{n,k}$, one can obtain a strongly consistent estimate of the log-likelihood by using

$$\hat{l}(\theta; g_{obs}) = -\log(|\mathcal{G}_{n,k}|) - \log \sum_{b=1}^B [\exp(\theta^T \Delta(G_b; g_{obs}))] + \log B \quad (5)$$

$$= -\log(|\mathcal{G}_{n,k}|) + \theta^T t(g_{obs}) - \log \hat{c}_k(\theta) + \log B, \quad (6)$$

where

$$\hat{c}_k(\theta) = \sum_{b=1}^B \exp(\theta^T t(G_b)). \quad (7)$$

The estimated log-likelihood in Equation (5) is maximized to obtain an approximate maximum likelihood estimator. Thus, the approximate MLE is defined as

$$\tilde{\theta} = \arg \max \hat{l}(\theta, g_{obs}) \quad (8)$$

Note that we don't need to compute $\log |\mathcal{G}_{n,k}|$, as it is a constant independent of θ .

To maximize the estimated log-likelihood in Equation (5), we use the Newton-Raphson method. This requires the first and second derivatives of the approximate log-likelihood $\hat{l}(\theta; g_{obs})$, which are easily calculated as follows. Let $t_j = t_j(g)$ be the j^{th} component of the sufficient statistic vector $t = t(g) = \{t_1(g), \dots, t_d(g)\}$. Let

$$\hat{\mu}_\theta(f(t)) = \sum_{b=1}^B f(t(G_b)) \exp(\theta^T t(G_b)).$$

Then, taking the derivatives of Equation (6) provides:

$$\frac{\partial \hat{l}}{\partial \theta_i} = t_i(g_{obs}) - \frac{\hat{\mu}_\theta(t_i)}{\hat{c}_k(\theta)} \quad (9)$$

and

$$\frac{\partial^2 \hat{l}}{\partial \theta_i^2} = -\frac{\hat{\mu}_\theta(t_i^2)}{\hat{c}_k(\theta)} + \frac{\hat{\mu}_\theta(t_i)^2}{\hat{c}_k(\theta)^2} \quad (10)$$

$$\frac{\partial^2 \hat{l}}{\partial \theta_i \partial \theta_j} = -\frac{\hat{\mu}_\theta(t_i t_j)}{\hat{c}_k(\theta)} + \frac{\hat{\mu}_\theta(t_i) \cdot \hat{\mu}_\theta(t_j)}{\hat{c}_k(\theta)^2}. \quad (11)$$

A note on the estimation of variance and Monte Carlo error is in order. There are two sources of error in the estimation of θ : the usual error of approximating the truth θ by the MLE $\hat{\theta}$ and the Monte Carlo error due to approximating $\hat{\theta}$ by $\tilde{\theta}$ from Equation (8). In most cases, the variance due to the first source is computed by appealing to a central limit theorem (CLT): Under regularity conditions, the MLE $\hat{\theta}$ converges to a normal distribution with mean θ and variance given by the inverse of the Fisher information matrix $I(\theta)$. However, application of a CLT to network models is incorrect, since the sample size of a network is 1. Even if we consider the number of nodes n to be the sample size, the observations are not independent and a CLT cannot be applied. There is no known solution to this problem, and most authors report the variance estimated by appealing to the CLT. On the other hand, the variance due to the Monte Carlo error can be computed by using a central limit theorem due to the fact that the likelihood is estimated using a Monte Carlo sample. The formulas for computing the variances from both of the sources can be found in [Geyer and Thompson \(1992\)](#) and [Hunter et al. \(2008\)](#) and are not repeated here.

In what follows, we discuss how the combination of using a support restriction and a uniform sampler (instead of sampling from the model using MCMC) improves the model behavior and thus improves the estimation. Simulations supporting these findings are presented in Section 3.

2.1. Existence of MLE and the approximate MLE

There are two likelihood functions: the true likelihood $l(\theta)$ given by Equation (3) and the estimated likelihood $\hat{l}(\theta)$ given by Equation (5). Correspondingly, there are two maximizers, the true MLE $\hat{\theta}$ and the approximate MLE $\tilde{\theta}$. We will discuss the existence of the true MLE and the approximate MLE and argue that using a smaller k makes the estimation of the MLE easier.

Using the standard theory of exponential families ([Barndorff-Nielsen, 2014](#)), existence of the true MLE $\hat{\theta}$ depends on the marginal polytope, that is, the convex hull of sufficient statistics of the set $\mathcal{G}_{n,k}$. The log-likelihood function is concave and a unique maximum exists if and only if the observed sufficient statistic $t(g_{obs})$ lies in the relative interior of the marginal polytope. The marginal polytopes of ERGMs are difficult to obtain in general (see for example [Engström and Norén \(2011\)](#)) and known only in few special cases, such as [Rinaldo, Petrović and Fienberg \(2013\)](#), [Karwa and Slavković \(2016\)](#). Obtaining the marginal polytopes for the degeneracy-restricted ERGMs appears to be more difficult and is an open problem in general, as it can only be computed for one specific DERGM at a time. We will compute these polytopes numerically for the edge-triangle DERGM in Section 3.

On the other hand, existence of the approximate MLE can be checked numerically. As discussed in [Handcock et al. \(2003\)](#), the estimated log-likelihood (5) can be written as the log-likelihood of a model from a discrete exponential family with support over $t(G_1), \dots, t(G_B)$ with observed sufficient statistic $t(g_{obs})$. Hence, using

again the standard theory of exponential families (Barndorff-Nielsen, 2014), one can show that the estimated log-likelihood is concave and Equation (5) has a unique maximum if and only if 0 lies in the interior of the convex hull of $\{\Delta(G_1), \dots, \Delta(G_B)\}$. Note that this polytope can be computed easily. To ensure that 0 lies in the interior of the convex hull of $\{\Delta(G_1), \dots, \Delta(G_B)\}$, one only needs to ensure that there exist graphs G_b and $G_{b'}$ such that $t_i(G_b) < t_i(g_{obs}) < t_i(G_{b'})$ for all $i = 1, \dots, d$.

Thus, assuming that the MLE exists, the existence of the approximate MLE is crucially tied to the sampling algorithm used to approximate the likelihood, which in turn depends on the behavior of the model. Next, we discuss why a uniform Monte Carlo sampler is better than a MCMC sampler from the model, followed by a discussion on how the bad behavior of ERGMs amplifies the issues with the MCMC sampler. The support restriction, along with a uniform sampler, naturally solves some of these issues.

2.2. Sampling from the model using MCMC vs. uniform sampling using MC

A common approach to approximate the likelihood function in Equation (3) is to consider its ratio at an arbitrary parameter value θ and a fixed parameter value θ_0 :

$$r(\theta, \theta_0) = \frac{l(\theta; g_{obs})}{l(\theta_0; g_{obs})}.$$

The ratio $r(\theta, \theta_0)$ is approximated by sampling a sequence of graphs G_1, \dots, G_B from an ERGM with parameter θ_0 , denoted by \mathbb{P}_{θ_0} . The approximated likelihood is maximized to obtain an approximate MLE $\hat{\theta}$, which is again used to sample graphs and find a better approximation and hence an improved MLE. The process is repeated until convergence. In general, since it is not possible to obtain iid samples from \mathbb{P}_{θ_0} , one resorts to MCMC methods to draw approximate samples from the model by running the Markov chain until convergence, see Snijders (2002) and Hunter and Handcock (2006) for more details.

Samples from \mathbb{P}_{θ_0} approximate $l(\theta)$ in a neighborhood of θ_0 . The accuracy of the estimated $l(\theta)$ decreases as θ moves further away from θ_0 . Moreover, the samples are correlated since the samples are generated using a Markov chain, and there may be convergence issues. On the other hand, a uniform Monte Carlo sampler generates graphs independent of any θ and hence – as long as the sample is large enough – $l(\theta)$ is well approximated for all values of θ .

One disadvantage of the uniform sampler is that one may need a large sample to obtain a good estimate of $l(\theta, g_{obs})$ due to the high-dimensional support of the model. On the other hand, an MCMC sampler from the model has at least three disadvantages from which a uniform Monte Carlo sampler does not suffer. First, one may need to iterate between maximizing the likelihood and sampling graphs. Second, the Markov chain may suffer from convergence issues which are difficult to diagnose. Third, even if the Markov chain converges, the samples are highly correlated. The disadvantages of using an MCMC sampler from the model are further exacerbated due to the bad behavior of the ERGMs, discussed next.

2.3. Choice of k and relation to existence of approximate MLE

As shown in Handcock et al. (2003), the difficulty in approximating the likelihood using MCMC samples from \mathbb{P}_{θ_0} is inherently tied to the ‘bad behavior’ of the ERGM. This ‘bad behavior’ refers to the following property: for a large subset of Θ , the model places most of its mass on very small regions of the support. Moreover, in many cases, these high probability regions of the support are *uninteresting* in the following sense: they correspond to graphs which are *extremal* with respect to the sufficient statistics, e.g. complete or empty graphs, or graphs corresponding to sufficient statistics that lie on or close to the boundary of the marginal polytope.

This behavior of the model amplifies the disadvantages of using MCMC samples from \mathbb{P}_{θ_0} as follows. Recall that the approximate MLE exists only when the convex hull of sufficient statistics sampled from \mathbb{P}_{θ_0} contains the observed sufficient statistic. Since the mass of \mathbb{P}_{θ_0} is concentrated on a very small region, unless θ_0 is close to the MLE, the sampled sufficient statistics will be very far from the observed sufficient statistics. Thus the existence of the approximate MLE depends crucially on the choice of θ_0 , see the discussion in Section 5.1 of Hunter et al. (2008) for an example. There have been several clever techniques proposed to solve this issue, such as moving θ_0 closer to $\hat{\theta}$ in steps, see Hummel, Hunter and Handcock (2012).

This issue is further complicated by the fact that the regions where \mathbb{P}_{θ_0} places most of its mass are disjoint subsets of extremal sufficient statistics, which are far from the observed sufficient statistics. Thus, even if we find a θ_0 close to the MLE, the Markov chain may suffer from convergence issues since it needs to move between high probability regions that are very far apart. By restricting the support of the model, we eliminate regions where the model places very little mass. This achieves two things: the model spreads the mass more uniformly and the likelihood is better behaved. (For illustrations, see Figures 1, 2, 3, 4.) Degeneracy serves as a natural support restriction, since it is a fundamental property of a network that controls its sparsity.

We arrive at the conclusion that DERGMs offer an improvement over ERGMs: restricting support eliminates the well-documented bad behavior of the model and, in turn, allows us to get a better estimate of the MLE. In Section 3, we illustrate these improvements numerically by studying the special case of the edge-triangle DERGM. Estimation of the parameters using Monte Carlo maximum likelihood relies crucially on the existence of an independent sampler from the set $\mathcal{G}_{n,k}$. Such an algorithm is discussed in Section 4.

3. Model behavior: simulations for the edge-triangle DERGM

To show computational evidence for the theoretical developments, we focus on the edge-triangle DERGM as a running example, a model whose sufficient statistics are the number of edges and the number of triangles of the graph. This model is also the running example in [Rinaldo, Fienberg and Zhou \(2009\)](#), where the authors discuss model degeneracy and show it is captured by polyhedral geometry of the model. They provide an in-depth study for model behavior on $n = 9$ nodes and thus give a specific context to reflect upon and compare ERGMs to DERGMs.

Simulation results in this section show that the difficulty of MLE computations increases with values of graph degeneracy parameter k . We also compare estimates on a smaller ($n = 18$) and a larger ($n = 500$) dataset and illustrate improved performance and model behavior. The smaller data set is the well-known Sampson Monastery Network from [Sampson \(1968\)](#), while the larger one is synthetic and was created randomly to illustrate scalability of the method; see Section 3.4. To illustrate the changing behavior of the degeneracy-restricted ERGMs, in each of the following examples we fix n and vary k from the observed value to the maximum $k = n - 1$.

3.1. Model behavior changes with k

Consider first the issue of existence of the approximate MLE. Recall from Section 2.1 that in the Monte Carlo estimation, the approximate MLE does not exist when the observed sufficient statistics lies outside of the convex hull of the estimated model polytope, which consists of sufficient statistics of sampled graphs from the model support $\mathcal{G}_{n,k}$. In DERGMs, this is more likely to happen when the degeneracy parameter k is high while the observed degeneracy is small. As an example, consider the Sampson monastery data [Sampson \(1968\)](#), in particular, the time period T4, available at [Batagelj and Mrvar \(2006\)](#). In this data set, $n = 18$ and observed graph degeneracy is $k = 3$. We show the location of its sufficient statistic vector in the estimated model polytopes for the edge-triangle DERGMs with varying k values. Figure 1 shows the convex hull of the graphs sampled (using Algorithm 1) from the uniform distribution on $\mathcal{G}_{n,k}$ for $n = 18$ and increasing k . When $k = 3$, the observed sufficient statistic lies well in the relative interior of the sampled sufficient statistics. On the other hand, when $k = 6$ and higher, the observed sufficient statistic lies well outside the convex hull; note that while the Figures here report the results for sample size 100,000, we have indeed verified that they remain the same even after increasing the sample size to $B = 1,000,000$.

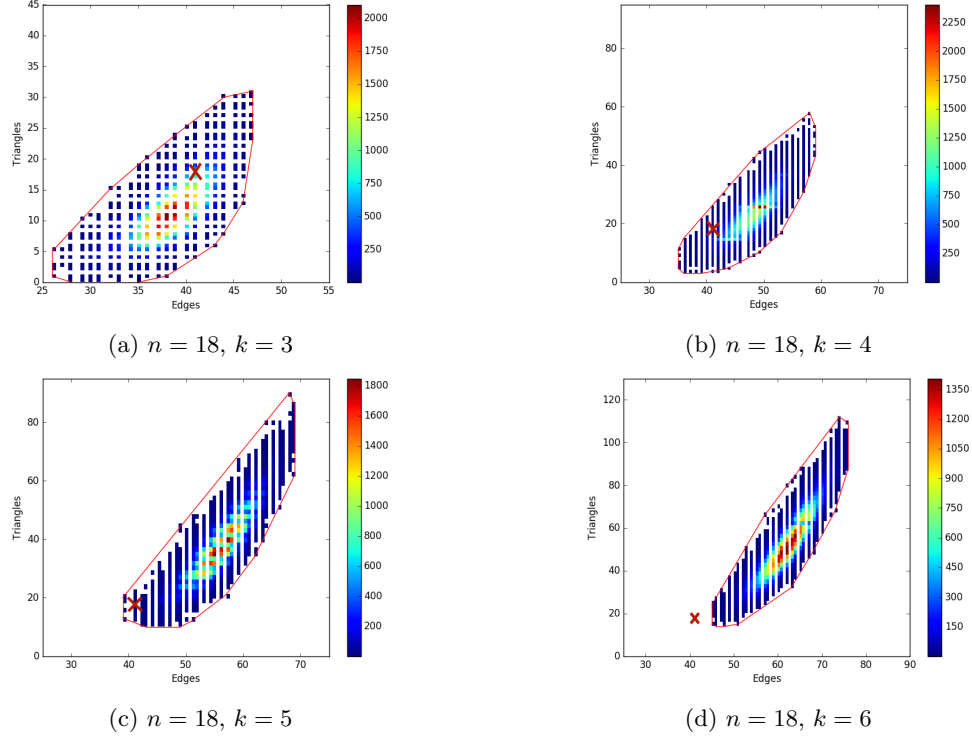


Fig 1: Estimated edge-triangle DERGM model polytopes for increasing k , with x marking the location of the observed sufficient statistic of the Sampson graph. Sample size is 100,000 each.

As k increases, the observed Sampson graph is progressively pushed out of the polytope. This is because for larger k , the model places more weight on denser graphs, making more sparse graphs such as the Sampson graph probabilistically less likely to appear. We do not plot the estimated polytopes for all other values of $k > 6$, but the reader can rest assured that the observed value of the sufficient statistics of the Sampson graph only gets farther removed from the convex hull.

A natural question arises from the discussion above: *how different* is the estimated polytope from the ‘true’ model polytope? Since the latter is actually only known for small values, specifically, $n = 9$, we can compare the estimated polytope of the DERGM with maximum degeneracy $k = 8$ to the complete support polytope for the edge-triangle ERGM, shown in [Rinaldo, Fienberg and Zhou \(2009\)](#). Note that Rinaldo et al. add, in the Discussion section, that it would take > 26 days to compute such a polytope on $n = 10$ nodes. Even with potential speed-ups that come with time, the enumeration of graphs for larger n is not scalable. Thus, we will answer the question on this toy example with $n = 9$ for comparison purposes.

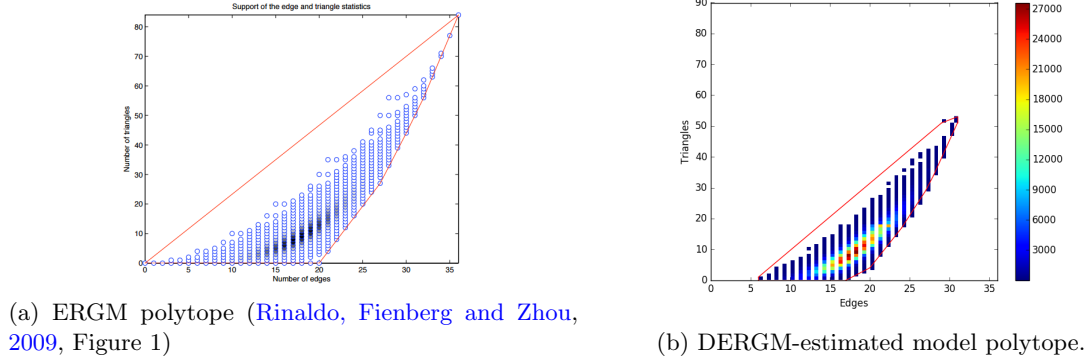


Fig 2: Comparison of sufficient statistics polytope from a complete ERGM of $\mathcal{G}_{9,8}$ with the polytope generated from random samples using the DERGM sampler Algorithm 1. Sample size: 1,000,000 graphs.

The estimated polytope of $\mathcal{G}_{9,8}$ is comparable to the complete polytope: the most dense region of the complete polytope also corresponds to the most dense region of the estimated polytope. However, the estimated polytope is missing some extremal graphs; this shows that these extremal graphs are probabilistically unlikely to be generated by the uniform sampler from the space of graphs $\mathcal{G}_9 = \mathcal{G}_{9,8}$.

As explained in Rinaldo, Fienberg and Zhou (2009) (see Section 3.4 therein for details), the shape of the estimated polytope above supports the argument that the original ERGM is ill-behaved. Specifically, they use Shannon's entropy, which captures the degree to which the model concentrates its mass on network configurations associated with a very small number of network statistics. The rationale is that degenerate models have large areas of low entropy. The correspondence between the model polytope and model degeneracy derived by Rinaldo et al. shows that the extremal rays of the normal fan of the model polytope correspond to directions of the ridges of Shannon's entropy function where it converges to some fixed value. These extremal rays are outer-normals of the facets (in our case, edges) of the polytope; we see that as k grows, the polytope becomes 'flatter' or, equivalently, the directions of the outer-normals of the edges on the lower hull get closer together, making the area of high entropy smaller. Although the exact plots are unavailable for the full ERGM on $n = 18$, we know that for $n = 9$ already the rays of normal fan concentrated in the small area of the space implying that the model has low entropy and is degenerate for a vast majority of parameter values; cf. (Rinaldo, Fienberg and Zhou, 2009, Figure 4A). As the authors there justify, we use the mean value parameters to illustrate this behavior, where it can be clearly seen.

To this end, Figure 3 shows that the higher-entropy region is more 'spread out' across the parameter region for the DERGMs with smaller values of k . While one cannot, of course, conclude that the model is non-degenerate for all possible parameter values, it is clear that the size of the parameter space that correspond to degenerate regions is certainly less than in the full ERGM. Regarding the caveat that the Figures are also estimated and not exact, we are nevertheless confident in the results, as 1) the algorithm used is a uniform sampler of well-ordered graphs from the model support $\mathcal{G}_{n,k}$ and 2) the estimated polytope is not far off from the true model polytope, at least as illustrated on the example above.

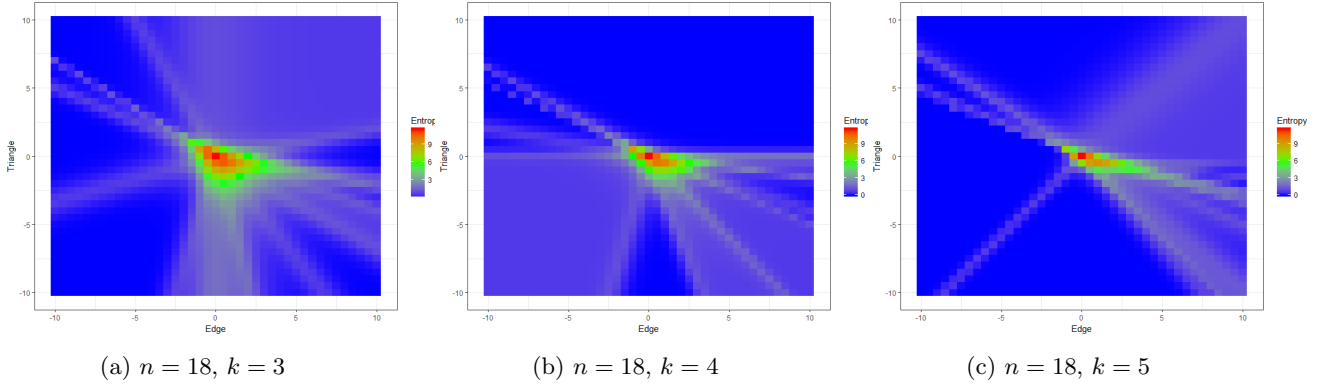


Fig 3: Comparison of degenerate (low-entropy) regions in the mean-value parameter space for the edge-triangle DERGMs on $n = 18$ and increasing k . Sample sizes are 100,000. As k increases, the high-entropy region becomes smaller.

3.2. The likelihood surface changes with k

As is well-known in the literature, choosing a starting point that is not near the true MLE can lead to non-convergence, or an unreliable estimate, in the MCMC-MLE algorithm for the usual ERGM (recall also our previous discussion in Section 2.3). In fact, most reasonable software packages such as R's `ergm` report that the estimation MCMC-MLE algorithm did not converge after a certain number of steps, and suggest trying a different starting point. While this behavior is documented for general ERGMs, we used Algorithm 1 to sample from the sample space of the model, both $\mathcal{G}_n = \mathcal{G}_{n,n-1}$ for the full ERGM and $\mathcal{G}_{n,k}$ with $k < n-1$ for various DERGMs, to estimate the likelihood function. Simulations uncover an interesting trend: the likelihood surface becomes ‘flatter’ around the maximum value as k grows, making it more difficult to find the maximum itself after a certain number of steps, and providing a reason for high sensitivity of the MCMC-MLE to the choice of the starting point. Figure 4 shows the contours of the (estimated) likelihood function at a fixed (but otherwise arbitrary) value $\theta = (\theta_1, \theta_2)$.

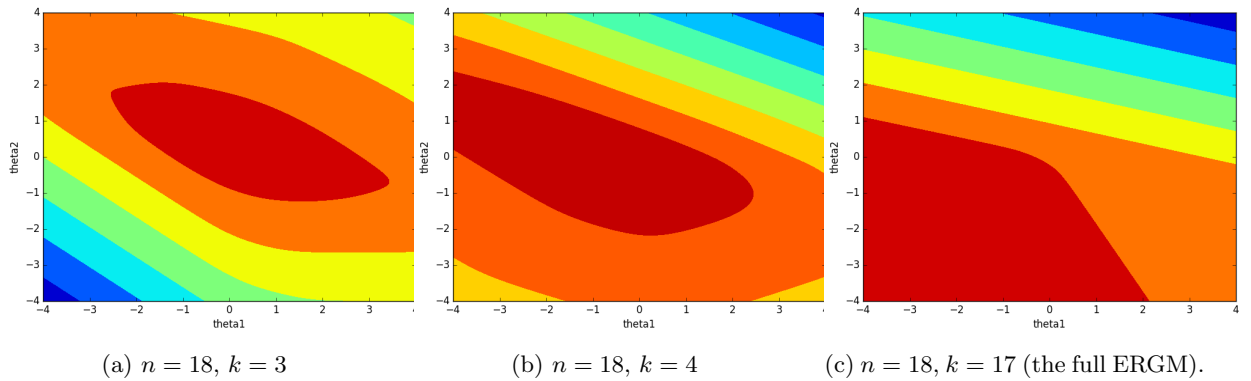


Fig 4: Contour plots of the estimated edge-triangle DERGM likelihood functions at various values of (θ_1, θ_2) for fixed n and increasing k . The estimated likelihood is based on an iid sample of 25,000 graphs in $\mathcal{G}_{n,k}$. The simulation was repeated for various (random) choices of (θ_1, θ_2) and the resulting plots showed exactly the same trend.

3.3. Choice of k

Choosing the value of k and thus selecting what DERGM to fit on a given data set is similar to the problem of model selection. Valid choices of k range from the observed value k_{obs} to $n - 1$, where $k = n - 1$ reduces to the usual ERGM. As we show by simulations throughout this Section, choosing smaller values of k can lead to a likelihood function that is better behaved. This is because a smaller k eliminates dense graphs from the support of the model and improves model degeneracy issues. Based on our results, one may wish to set k to be the observed value k_{obs} . Values that are larger and closer to the number of nodes n tend to lead to more degenerate distributions.

3.4. Estimation and fitting the edge-triangle DERGM

We revisit the Sampson dataset from Section 3.1 with $n = 18$ nodes and $m = 41$ edges and also consider a synthetic data set with $n = 500$ nodes and $m = 2460$ edges. We name the latter the Ersatz Friend Network; edge list is available upon request. In what follows we compare the performance of estimation/analyses using the edge triangle DERGM against the standard edge-triangle ERGM. In both cases, we set $k = k_{obs}$.

The Sampson network has degeneracy $k = 3$. Thus we simulate $B = 100,000$ graphs from $\mathcal{G}_{18,3}$. To estimate the MC-MLE, we choose random starting points for each simulation and return the computed MLE via the Newton-Rhaphson method.

Simulation 1:

	MCMLE	stddev	MCErrror
[1,]	-0.1384894	0.5778615	1.202384e-06
[2,]	0.3758031	0.3110103	3.925308e-07

Simulation 2:

	MCMLE	stddev	MCErrror
[1,]	-0.1384875	0.5778618	1.202385e-06
[2,]	0.3758024	0.3110104	3.925310e-07

Simulation 3:

	MCMLE	stddev	MCErrror
[1,]	-0.1384924	0.5778612	1.202393e-06
[2,]	0.3758051	0.3110102	3.925337e-07

Using the DERGM with $k = 3$, the MC-MLE converges to a stable solution for each of the simulations we ran; we report three of them above. The Monte Carlo sampler relies on Algorithm 1. Note that for $k = 3$, the observed network is inside the convex hull of the estimated model polytope for the DERGM. Therefore, the MC-MLE results are accurate and not sensitive to initial starting points. This indicates good behavior.

Meanwhile, R's `ergm` MCMC-MLE estimate returned the following:

This model was fit using MCMC. To examine model diagnostics and check for degeneracy, use the `mcmc.diagnostics()` function.

MCMC sample of size based on:

edges	triangle
-1.1817	0.1473

Monte Carlo MLE Coefficients:

edges	triangle
-1.1946	0.1538

As the Sampson Network is a relatively small dataset, the usual ERGM converges to a result. This result differs from the MC-MLE estimate; this could be due to the fact that the models we are fitting differ, but also note that the observed Sampson network's sufficient statistics are far removed from the center of the convex hull of the model polytope for the full ERGM, as was illustrated Figure 1. Although the standard ERGM's MCMC-MLE converged, we cannot be sure that it is reliable due to the shape of the likelihood.

Next we examine the Ersatz Friend Network. This network has $n = 500$ nodes, $m = 2460$ edges, and degeneracy $k_{obs} = 5$. Even though it is a synthetic network, its low degeneracy mirrors what is seen in real-world networks; again, compare to (Karwa et al., Sec. 3.1). Using the MC sampler from Algorithm 1, which scaled reasonably well to 500 nodes, we simulate $B = 25,000$ graphs from $\mathcal{G}_{500,5}$. The location of the Ersatz network’s observed edge-triangle pair in the convex hull of the simulated sufficient statistics (that is, the estimated model polytope) is shown in Figure 5.

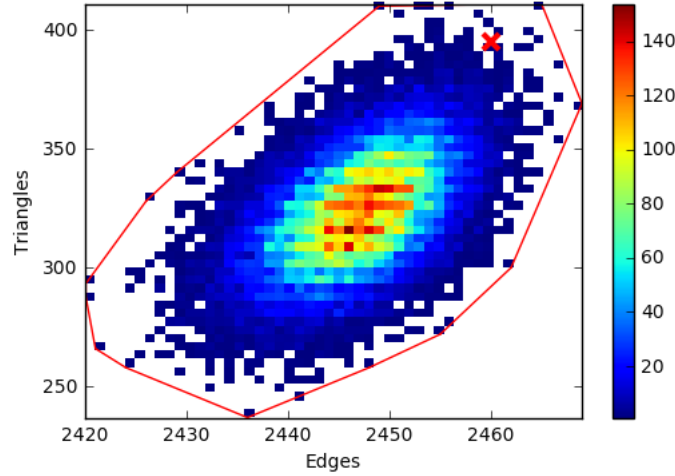


Fig 5: Estimated edge-triangle DERGM model polytope for $\mathcal{G}_{500,5}$. Sample size 25,000. The Ersatz Friend Network’s edge-triangle vector is marked by an x and lies within the convex hull.

As the observed edge-triangle vector lies inside the polytope, the estimation for the MLE $\hat{\theta}$ should converge to a stable solution. Indeed, the results for the MC-MLE for the edge-triangle DERGM are as follows.

MC-MLE Simulations

Simulation 1:

	MCMLE	stddev	MCErrror
[1,]	0.1995317	0.2247923	0.0013338296
[2,]	0.1337830	0.0674917	0.0001344955

Simulation 2:

	MCMLE	stddev	MCErrror
[1,]	0.1995487	0.22479778	0.0013339343
[2,]	0.1337800	0.06749126	0.0001345075

Simulation 3:

	MCMLE	stddev	MCErrror
[1,]	0.1995485	0.22479772	0.0013339322
[2,]	0.1337800	0.06749125	0.0001345072

Meanwhile, the results for the MCMC-MLE for the corresponding edge-triangle ERGM are not so great:

MCMC-MLE Simulations

```
Error in ergm.MCMLE(init, nw, model, initialfit = (initialfit <- NULL), :
Number of edges in a simulated network exceeds that in the observed by a factor of more than 20.
This is a strong indicator of model degeneracy or a very poor starting parameter configuration.
If you are reasonably certain that neither of these is the case, increase the
MCMLE.density.guard control.ergm() parameter.
```

In the situation similar to the Sampson network, the Ersatz Friendship network was captured in the DERGM-estimated polytope, thus leading the MC-MLE to converge to a stable solution regardless of a starting point. This was not the case for the MCMC-MLE: it failed to converge to a solution during 90% of the runs in an experiment consisting of 10 distinct runs with varying starting points. This again verifies high sensitivity to the initial starting point as the usual ERGM does not encapsulate the observed network; this further elucidates the ill-behavior of the standard ERGM. While there are methods to try to improve the starting point for the MCMC-MLE (see discussion in Section 2.3), the DERGM algorithm does not require such manipulations.

4. Uniform samplers for $\mathcal{G}_{n,k}$

The main contribution of this section is the development of a fast uniform sampler of the space of well-ordered graphs in $\mathcal{G}_{n,k}$, contained in Section 4.1, which has been used throughout Section 3 in simulations, both for MLE estimation and estimated polytope plots. We discuss the basis of the algorithm and the updates we made to make it scalable. This algorithm can be used stand-alone for Monte Carlo sampling for DERGM estimation, specifically in the case when non-well-ordered graphs are not of interest. On the other hand, it can also be used in combination with a non-well-ordered sampler to create a stratified sampler for all graphs of $\mathcal{G}_{n,k}$ when needed; below, we discuss how sometimes the stratified sampler effectively reduces to the well-ordered one. Finally, if the observed graph is well outside the convex hull of sampled graphs, one may wish to use a fast importance MCMC sampler, in conjunction with the uniform sampler from Section 4.1 to create an umbrella sampler on $\mathcal{G}_{n,k}$. The umbrella sampler converged quickly in simulations, but we omit those results here as they were not necessary for the data sets we analyze.

4.1. A uniform sampler for well-ordered graphs from $\mathcal{G}_{n,k}$

In (Bauer, Krug and Wagner, 2010, Algorithm 1), the authors derive a uniform sampler for the set of well-ordered graphs in $\mathcal{G}_{n,k}$. A *well-ordered* graph is one in which the node labels are ordered so that no vertex has more than k neighbors with a higher label.

Using this algorithm as a starting point, we make several key changes to ensure that their algorithm is computationally efficient: we convert their algorithm from a recursive one to an iterative one. By doing this, we eliminate many complexity problems inherent in the original algorithm. Specifically, the iterative version eliminates stack overflow issues for large graphs, as well as greatly reduces the execution time of generating a graph.

Let us take a closer look at the improved, scalable version of (Bauer, Krug and Wagner, 2010, Algorithm

1), in which we additionally fix some typographical errors.

Algorithm 1: Generate a well-ordered g from $\mathcal{G}_{n,k}$ uniformly.

```

input :  $n$ , the number of nodes,
         $k$ , maximum graph degeneracy.
output:  $g$ , a graph in  $\mathcal{G}_{n,k}$  in which every vertex  $i$  has no more than  $\geq k$  neighbors in the set  $\{i+1, \dots, n\}$ .

1 for  $i = 1$  to  $n$  do
2    $d_i \sim \text{restrictedBinomial}(n - i, \min(n - i, k))$ 
3   if  $i = n$  then
4      $V = V \cup \{n\}$ 
5   end
6 end
7 for  $i = n$  to  $1$  do
8    $T = \{\}$ 
9    $P = V$ 
10   $a = |P|$ 
11  for  $j = 0$  to  $d_i - 1$  do
12     $m \sim \text{Uniform}(0, a - j)$ 
13     $T = T \cup \{(i, P_m)\}$ 
14     $P_m = P_{a-j-1}$ 
15  end
16   $V = V \cup \{i\}$ 
17   $E = E \cup \{T\}$ 
18 end
19  $G = \{V, E\}$ 
20 return  $G$ 

```

Recursion is emulated using two for-loops. The first for-loop populates a list of degrees where each index of the list corresponds to the respective vertex label. The degrees for each vertex are generated using a restricted binomial distribution. Instead of utilizing the cumulative distribution and using binary search to obtain values as suggested by the original paper, we opt to use the probability density function and store the values in a list data structure, reducing the complexity of obtaining the degree values. When the loop reaches the very last vertex, we add that vertex to the working vertex set. For each iteration in the second for-loop, a temporary copy of the current working vertex set is created. We then uniformly generate d_i indices to sample without replacement from the vertex set copy, and use these samples for the edge set of the current vertex. It is obvious that this sample is uniformly generated, complying with the original algorithm.

For a benchmark, we tested the original recursive version (including generating all possible combinations) and the new iterative version on a machine with the following specifications: Intel Core i7-4790K CPU @ 4.00 GHz, 8 GB DDR3 RAM, Arch Linux x64, with the following results:

(n,k)	Original Recursive Algorithm	Final Iterative Algorithm 1
(50,8)	3.96 seconds	0.03 seconds
(800,2)	Stack Overflow	0.51 seconds
(3000,2)	Stack Overflow	1.90 seconds

(Note: Times will vary slightly based on your computer's hardware.)

The results clearly indicate that the scalable version is superior in regards to time complexity.

In some applications, it may be desirable to further restrict the sample space of the model by restricting the total number of edges of the graph, or use such a restriction for stratified sampling of $\mathcal{G}_{n,k}$. To that end, let $\mathcal{G}_{n,m,k}$ be the set of graphs on n nodes and degeneracy k with exactly m edges. (Bauer, Krug and Wagner, 2010, Algorithm 2) offer an algorithm for uniform sampling of $\mathcal{G}_{n,m,k}$, however, it was not implemented due to the complexity of step 3 that the authors suggest be implemented using Equation (2.7) in Bauer, Krug and Wagner (2010). Pre-computation of degrees proved nearly impossible in practice for several reasons. The recursive nature of calculating the cardinality for possible graphs of given vertices, edges, and degeneracy yielded very inefficient computations in which the run time of each computation was longer than trying to generate whole graphs by other means. While we were able to alleviate this issue somewhat by utilizing a dynamic programming approach with memoization, even for semi-sparse, average size graphs, numerical

overflow occurred, which rendered the speed increase fruitless. Instead, we opt to use (Bauer, Krug and Wagner, 2010, Algorithm 3), which is a non-uniform but fast sampler of $\mathcal{G}_{n,m,k}$. Our implementation of this algorithm, outlined in Algorithm 2, stays true to the pseudo-code given in the original paper, with the only alteration being utilizing the same approach to uniform selection as in our implementation of Algorithm 1.

Algorithm 2: Generate a well-ordered g from $\mathcal{G}_{n,m,k}$ non-uniformly.

```

input :  $n$ , the number of nodes,
         $m$ , the number of edges,
         $k$ , maximum graph degeneracy.
output:  $g$ , a graph in  $\mathcal{G}_{n,k}$  with  $m$  edges in which every vertex  $i$  has no more than  $\geq k$  neighbors in the set  $\{i+1, \dots, n\}$ .

1  $C = 1, \dots, v_{n-1}$ 
2 for  $i = 1$  to  $m$  do
3    $j \sim \text{Uniform}(0, |C|)$ 
4    $d_j = d_j + 1$ 
5   if  $d_j = \min(n - v_j, k)$  then
6      $C \leftarrow C \setminus \{v_j\}$ 
7   end
8 end
9 for  $i = 1$  to  $n - 1$  do
10   $T = \{\}$ 
11   $P = V$ 
12   $a = |P|$ 
13  for  $j = 0$  to  $d_i - 1$  do
14     $m \sim \text{Uniform}(0, a - j)$ 
15     $T = T \cup \{(i, P_m)\}$ 
16     $P_m = P_{a-j-1}$ 
17  end
18   $V = V \cup \{i\}$ 
19   $E = E \cup \{T\}$ 
20 end
21  $G = \{V, E\}$ 
22 return  $G$ 

```

4.2. Stratified sampling of $\mathcal{G}_{n,k}$ to include non-well-ordered graphs if needed

Another issue with (Bauer, Krug and Wagner, 2010, Algo.1) is that it generates only so-called ‘well-ordered’ graphs in $\mathcal{G}_{n,k}$. This misses a part of graphs in the support of our model. To remedy this issue, we classify all missing graphs and produce them via stratified sampling with two strata. Specifically, Algorithm 1 is used to sample from the set of well-ordered graphs in $\mathcal{G}_{n,k}$, while Algorithm 3, described below, is used to generate non-well-ordered graphs in $\mathcal{G}_{n,k}$. Let n_1 and n_2 be the number of well-ordered and non-well-ordered graphs, respectively. The formula for n_1 is provided in Bauer, Krug and Wagner (2010) under the notation $D_n^{(k)}$, while n_2 is studied below. To the best of our knowledge, the literature does not provide a good estimate of the number n_1 of well-ordered k -degenerate graphs compared to the total number of k -degenerate graphs. It should be noted that, in practice, the uniform sampler from Section 4.1 may only be omitting a tiny fraction of graphs in the support of the DERGM; this situation is described in detail at the end of this Section. Therefore, the reader interested in applications more than in theory behind the algorithms that may not be necessary in practice may skip the remainder of this technical section.

A graph $g \in \mathcal{G}_{n,k}$ is *not well-ordered* if there exists at least one vertex j with at least $k+1$ neighbors in the set $\{j+1, \dots, n\}$. Among all such vertices with too many big neighbors, let $k+c$ be the minimum such number of big neighbors, and let i be the index of the smallest vertex that has $k+c$ big neighbors. We construct non-well-ordered graphs and use them to estimate n_1 by going through possible cases for the values of c and i . For each case $c = 1, \dots, n-k-1$, some vertex i has $k+c$ neighbors in the set $\{i+1, \dots, n\}$. For each of the cases, the vertex i can be chosen from the set $\{1, \dots, n-(k+c)\}$. Note that these $k+c$ neighbors of i can be connected in any arbitrary way, as long as the entire graph is in $\mathcal{G}_{n,k}$. Thus, we proceed as follows: construct a random graph h on $k+c$ vertices whose labels are in the set $\{i+1, \dots, n\}$. Then, construct a suspension g over h using vertex i , that is, ensure that i is connected to all $k+c$ vertices of h . Finally, the vertices $\{1, \dots, i\}$ can be connected in any way such that, by minimality of i , the resulting subgraph on $\{1, \dots, i\}$ is well-ordered

and, additionally, each vertex in the set $\{1, \dots, i\}$ can have at most k neighbors in the vertex set $\{i+1, \dots, n\}$. The construction is outlined in Algorithm 3.

Algorithm 3: Generate a non-well-ordered g from $\mathcal{G}_{n,k}$

- input** : n , the number of nodes,
 k , maximum graph degeneracy.
output: g , a graph in $\mathcal{G}_{n,k}$ (or $\mathcal{G}_{n,d}$ with $d > k$, unfortunately) in which there is a vertex i that has $\geq k+1$ neighbors in the set $\{i+1, \dots, n\}$.
- 1 Pick $c \in \{1, \dots, n-k-1\}$.
 - 2 Pick $i \in \{1, \dots, n-(k+c)\}$.
 - 3 Use Algorithm 1 to sample $\tilde{h} \in \mathcal{G}_{k+c, k+c-1}$; repeat until $\text{degen}(\tilde{h}) \leq k$.
 - 4 Choose (uniformly) a subset of $k+c$ vertex labels from the set of legal vertex labels $\{i+1, \dots, n\}$.
 - 5 Let h be the graph obtained from \tilde{h} by replacing the labels $1, \dots, k$ by those selected on Line 4.
 - 6 Create the suspension graph g over h by adding to h edges $\{i, x\}$ for all $x \in V(h)$.
 - 7 Connect vertices $\{1, \dots, i\}$ by constructing any well-ordered graph from $\mathcal{G}_{i,k}$.
 - 8 Connect any of the vertices $\{1, \dots, i\}$ to at most k vertices in the set $\{i+1, \dots, n\}$.
 - 9 Output g if $\text{degen}(g) \leq k$; otherwise return to Step 1.
-

There are $\binom{n-i}{k+c}$ ways to choose the neighbors of the vertex i on Line 4 and for each choice of neighbors there are $2^{\binom{k+c}{2}}$ graphs \tilde{h} generated on Line 3. There are $D_i^{(k)}$ well-ordered graphs on Line 7 and $i \sum_{p=1}^k \binom{n-i}{p}$ graphs on Line 8. Thus, Algorithm 3 constructs the following number of graphs g :

$$\underbrace{\sum_{i=1}^{n-(k+1)} \underbrace{\binom{n-i}{k+1}}_{\text{Line 4}} \cdot \underbrace{2^{\binom{k+1}{2}}}_{\text{Line 3}} \cdot \underbrace{D_i^{(k)}}_{\text{Line 7}} \cdot \underbrace{i \sum_{p=1}^k \binom{n-i}{p}}_{\text{Line 8}}}_{c=1} + \underbrace{\sum_{i=1}^{n-(k+2)} \binom{n-i}{k+2} \cdot 2^{\binom{k+2}{2}} \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p}}_{c=2} + \dots$$

$$\dots + \underbrace{\sum_{i=1}^{n-(k+n-k-1)} \binom{n-i}{n-1} \cdot 2^{\binom{n-1}{2}} \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p}}_{c=n-k-1} = \quad (12)$$

$$2^{\binom{k+1}{2}} \cdot \sum_{i=1}^{n-(k+1)} \binom{n-i}{k+1} \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p} + 2^{\binom{k+2}{2}} \cdot \sum_{i=1}^{n-(k+2)} \binom{n-i}{k+2} \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p} + \dots$$

$$\dots + 2^{\binom{n-1}{2}} \cdot \binom{n-1}{n-1} \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p}, \quad (13)$$

where each of the $n-k-1$ summands corresponds to one of the cases c .

Note that Equation (13) is an upper bound on n_2 , since it counts all graphs g constructed by Algorithm 3. It is also a strict upper bound on the number of graphs g actually returned by the algorithm, since it counts those graphs whose degeneracy happens to be strictly larger than k . Equation (13) counts all graphs on $k+c$ nodes, $2^{\binom{k+c}{2}}$, constructed in Step 3. Surely, a better count can be obtained by replacing $2^{\binom{k+c}{2}}$ by $2^{\binom{k+c}{2}} - \#\{\text{well-ordered graphs on } k+c \text{ vertices of degeneracy } > k\}$. Doing this replacement in the equation is, crucially, still an upper bound on n_2 (since the well-ordered graphs of degeneracy larger than k certainly do not contribute to any non-well-ordered graphs of degeneracy at most k). Since

$$\begin{aligned} & \#\{\text{well-ordered graphs on } k+c \text{ nodes of degeneracy } > k\} \\ &= \#\{\text{all well-ordered graphs on } k+c \text{ nodes except those of degeneracy } \leq k\} \\ &= D_{k+c}^{(k+c-1)} - D_{k+c}^{(k)}, \end{aligned}$$

the following is a better upper bound on the number of graphs we wish to keep from Algorithm 3 and thus

also an upper bound on n_2 :

$$\begin{aligned}
& \sum_{i=1}^{n-(k+1)} \binom{n-i}{k+1} \cdot \left(2^{\binom{k+1}{2}} - \left(D_{k+1}^{(k+1-1)} - D_{k+1}^{(k)} \right) \right) \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p} + \\
& \sum_{i=1}^{n-(k+2)} \binom{n-i}{k+2} \cdot \left(2^{\binom{k+2}{2}} - \left(D_{k+2}^{(k+2-1)} - D_{k+2}^{(k)} \right) \right) \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p} + \dots \\
& \dots + \binom{n-1}{n-1} \cdot \underbrace{\left(2^{\binom{n-1}{2}} - \left(D_{n-1}^{(n-1-1)} - D_{n-1}^{(k)} \right) \right)}_{\text{Line 3 minus well-ordered of } \text{degen}_{>k}} \cdot D_i^{(k)} \cdot i \sum_{p=1}^k \binom{n-i}{p}. \tag{14}
\end{aligned}$$

Let

$$t_{true} = \log n_1 / (n_1 + n_2)$$

be the true threshold used to divide the sample in two strata and define

$$t_{estimated} = \log n_1 / (n_1 + (14)).$$

Given that $(14) > n_2$, $t_{estimated} < t_{true} \leq 0$. Therefore we take the following approach: 1) compute the threshold $t_{estimated}$ for the fixed n and k for which we wish to run the current simulation. 2) If $t_{estimated}$ is close to 0, then that forces t_{true} to be close to 0, which in turn means that there is a very, very small number of non-well-ordered graphs for that choice of n and k and therefore the stratified sampler essentially reduces to sampling well-ordered graphs only.

Of course, if $t_{estimated}$ is not relatively close to 0, then for those values of n and k , while it is possible that t_{true} is close to 0, one should implement both the well-ordered and non-well-ordered algorithm. Falling back on the well-ordered algorithm is equivalent to using an approximate sampler in practice. The users may additionally prefer to replace Algorithm 3 by instead permuting the vertices of the output of Algorithm 1, allowing it to reach the entire sample space $\mathcal{G}_{n,k}$ in another way.

In practice, if the model's sufficient statistics are subgraph counts (or if the distribution is exchangeable), well-ordering does not pose a restriction, because in the uniform sampling using MC in estimating the MLE, only the values of the sufficient statistics of the sampled graphs are used. These are oblivious to vertex labels, so ordering is irrelevant.

5. Sampling From the Model

In this section, we briefly present an MCMC algorithm for sampling graphs from the DERGM for a fixed value of θ . Sampling from the model allows one to perform heuristic goodness-of-fit tests. One can also generate many graphs similar to an observed graph by estimating an MLE and sampling from the DERGM with parameter values $\hat{\theta}_{MLE}$.

Algorithm 4 to sample from a DERGM is a straightforward MCMC algorithm where the proposal is the uniform distribution from $\mathcal{G}_{n,k}$. This proposal ensures that the graphs being proposed are in the set $\mathcal{G}_{n,k}$. Let $\pi(g) \propto \exp(\theta^t t(g))$. The Metropolis-Hastings acceptance ratio becomes

$$\alpha(g_{current}, g_{proposed}) = \min \left(1, \frac{\pi(g_{proposed})}{\pi(g_{current})} \right)$$

Algorithm 4: Independent Metropolis algorithm to sample from the model

-
- input** : g_0 , the starting value of the chain
- 1 Let g_0 be the starting value of the chain and set $g_{\text{current}} = g_0$.
 - 2 For $t = 1, \dots, B$.
 - 3 Propose a new value g_{proposed} from $\mathcal{U}_{n,k}$
 - 4 Define

$$\alpha(g_{\text{current}}, g_{\text{proposed}}) = \min \left(1, \frac{\pi(g_{\text{proposed}})}{\pi(g_{\text{current}})} \right)$$

- 5 Let $u \sim \text{Unif}(0, 1)$.
 - 6 If $u \leq \alpha$, accept the new proposal and set $g_{t+1} = g_{\text{proposed}}$
 - 7 Else set $g_t = g_{\text{current}}$
-

An alternative algorithm is to use a symmetric proposal distribution with Metropolis Hastings acceptance ratio. For instance, one can use the Tie no-Tie proposal (see, for example, [Caimo and Friel \(2011\)](#)). Such proposals may generate graphs outside the set $\mathcal{G}_{n,k}$, which are naturally rejected by the Metropolis Hastings algorithm.

6. Discussion

We introduced a general modification of exponential family random graph models that solves the intractability and degenerate behavior issues. This modification amounts to a support restriction, by conditioning on the observed network's graph-degeneracy, which is a measure of sparsity that is weaker than a general bound on node degrees. The resulting model class, which we name *degeneracy-restricted* or *DERGMs*, does not suffer from the same estimation issues as the usual ERGMs. Specifically, the first advantage that DERGMs with smaller graph degeneracy parameter k offer is that they have a better-behaved simulated likelihood (i.e., more steep around the maximum) and the simulated model polytope spreads more mass around the model support, eliminating very low-probability extreme graphs in the model. This also makes estimation algorithms more stable.

The second advantage that DERGMs offer over ERGMs is that they come equipped with fast sampling algorithms, allowing a replacement of the usual MCMC-MLE that heavily depends on the starting point with an MC-MLE algorithm that does not. Of course, by letting the degeneracy parameter be the maximum possible (i.e., the number of nodes minus 1), this implies that we also can use the same Monte Carlo sampler for the usual ERGM; it also does not require a good starting point. However, since this is clearly not the only problem with ERGMs and their 'bad behavior', models with lower degeneracy parameter show further advantages. One of the threads of future work that promises to further improve the present considerations is the development of a distributed version of Algorithm 1. While we did run the current implementation in parallel, it can further be improved to run on a cluster. The current implementation scales very well to hundreds of nodes and with the additional step it should perform just as well on thousands.

The particular example of the edge-triangle DERGM presented here is a good illustration of the general DERGM behavior. It is a natural choice of the running example, given the recent work by [Rinaldo, Fienberg and Zhou \(2009\)](#) that studies its degenerate behavior in detail. The general framework presented, however, applies to any ERGM; a good overview of many of the popular classes being offered in [Goldenberg et al. \(2009\)](#). Recent work on the shell-distribution ERGM [Karwa et al.](#) introduces a limited version of the current contribution: it is an example of an ERGM with similarly restricted support and gives direct motivation for the study of DERGMs in general. However, there, the model support was not $\mathcal{G}_{n,k}$ for fixed n and k , but rather $\mathcal{G}_{n,k} \setminus \mathcal{G}_{n,k-1}$ - networks with degeneracy exactly k . Here we propose to use networks of degeneracy at most k , to enlarge the model support, and offer greater flexibility in modeling. Our contributions indicate that, while ERGMs are known to have difficulties, DERGMs may offer a feasible and interpretable modification of this powerful and flexible model class.

References

- BAJIĆ, D. (2016). DERGMs: Supplementary material on GitHub. <https://github.com/dbajic/degen>.
- BANNISTER, M. J., DEVANNY, W. E. and EPPSTEIN, D. (2014). ERGMs are Hard. Preprint arXiv:1412.1787 [cs.DS].
- BARNDORFF-NIELSEN, O. (2014). *Information and exponential families in statistical theory*. John Wiley & Sons.
- BATAGELJ, V. and MRVAR, A. (2006). Pajek datasets. <http://vlado.fmf.uni-lj.si/pub/networks/data/>.
- BAUER, R., KRUG, M. and WAGNER, D. (2010). Enumerating and Generating Labeled k -degenerate Graphs. In *Proceedings of the Seventh Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*.
- CAIMO, A. and FRIEL, N. (2011). Bayesian inference for exponential random graph models. *Social Networks* **33** 41–55.
- CHATTERJEE, S. and DIACONIS, P. (2013). Estimating and understanding exponential random graph models. *Annals of Statistics* **41** 2428–2461.
- ENGSTRÖM, A. and NORÉN, P. (2011). Polytopes from subgraph statistics. *Discrete Mathematics and Theoretical Computer Science*.
- FRANK, O. and STRAUSS, D. (1986). Markov graphs. *Journal of the American Statistical Association* **81** 832–842.
- GEYER, C. J. and THOMPSON, E. A. (1992). Constrained Monte Carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)* 657–699.
- GOLDENBERG, A., ZHENG, A. X., FIENBERG, S. E. and AIROLDI, E. M. (2009). A Survey of Statistical Network Models. *Foundations and Trends in Machine Learning* **2** 129–233.
- GOODREAU, S. M., KITTS, J. A. and MORRIS, M. (2009). Birds of a feather, or friend of a friend? using exponential random graph models to investigate adolescent social networks*. *Demography* **46** 103–125.
- HANDCOCK, M. S., ROBINS, G., SNIJDERS, T., MOODY, J. and BESAG, J. (2003). Assessing degeneracy in statistical models of social networks. *Center for Statistics and the Social Sciences, University of Washington, Working Paper No. 39*.
- HUMMEL, R. M., HUNTER, D. R. and HANDCOCK, M. S. (2012). Improving simulation-based algorithms for fitting ERGMs. *Journal of Computational and Graphical Statistics* **21** 920–939.
- HUNTER, D. R. and HANDCOCK, M. S. (2006). Inference in Curved Exponential Family Models for Networks. *Journal of Computational and Graphical Statistics* **15** 565–583.
- HUNTER, D. R., HANDCOCK, M. S., BUTTS, C. T., GOODREAU, S. M. and MORRIS, M. (2008). ergm: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of statistical software* **24** URL <http://www.jstatsoft.org/v24/i03>.
- KARWA, V. and SLAVKOVIĆ, A. (2016). Inference using noisy degrees: Differentially private β -model and synthetic graphs. *The Annals of Statistics* **44** 87–112.
- KARWA, V., PELSMAJER, M. J., PETROVIĆ, S., STASI, D. and WILBURNE, D. Statistical models for cores decomposition of an undirected random graph. Preprint, arXiv:1410.7357 [math.ST], in revision.
- RINALDO, A., FIENBERG, S. E. and ZHOU, Y. (2009). On the Geometry of Discrete Exponential Families with Application to Exponential Random Graph Models. *Electronic Journal of Statistics* **3** 446–484.
- RINALDO, A., PETROVIĆ, S. and FIENBERG, S. E. (2013). Maximum likelihood estimation in the β -model. *The Annals of Statistics* **41** 1085–1110.
- SAMPSON, S. F. (1968). A novitiate in a period of change: An experimental and case study of relationships PhD thesis, Department of Sociology, Cornell University.
- SAUL, Z. M. and FILKOV, V. (2007). Exploring biological network structure using exponential random graph models. *Bioinformatics* **23** 2604–2611.
- SEIDMAN, S. B. (1983). Network structure and minimum degree. *Social Networks* **5** 269–287.
- SNIJDERS, T. A. (2002). Markov chain Monte Carlo estimation of exponential random graph models. *Journal of Social Structure* **3** 1–40.
- SNIJDERS, T. A., PATTISON, P. E., ROBINS, G. L. and HANDCOCK, M. S. (2006). New specifications for exponential random graph models. *Sociological methodology* **36** 99–153.
- YIN, M., RINALDO, A. and FADNAVIS, S. (To appear). Asymptotic quantization of exponential random graphs. *Annals of Applied Probability*.

7. Appendix A: Some computational considerations

This section contains a useful implementation detail that helps deal with numerical issues common to ERGMs in general. The trick is standard but we review it here for completeness and we use it repeatedly in implementation in cases of numerical overflow.

7.1. Log-Sum-Exp Manipulation

While the algorithms described in Section 4 are efficient generators of iid graphs from $\mathcal{G}_{n,k}$, there is the additional computational hurdle of estimating the normalizing constant, which tends to be very large. To circumvent this difficulty, we utilize an approach popularized in Machine Learning called Log-Sum-Exp Manipulation, outlined here for completeness.

Suppose we wish to compute $\log(\sum_i \exp(a_i))$, where a_i 's can be large. To avoid numerical overflow, we let $b = \max_i a_i$, and compute the following:

$$\begin{aligned}
 \log\left(\sum_i \exp(a_i)\right) &= b + \log\left(\sum_i \exp(a_i - b)\right) \\
 &= b + \log\left(\sum_i \frac{\exp(a_i)}{\exp(b)}\right) \\
 &= b + \log\left(\frac{\sum_i \exp(a_i)}{\exp(b)}\right) \\
 &= b + \log\left(\sum_i \exp(a_i)\right) - \log(\exp(b)) \\
 &= b + \log\left(\sum_i \exp(a_i)\right) - b \\
 &= \log\left(\sum_i \exp(a_i)\right)
 \end{aligned}$$

The benefit of Log-Sum-Exp Manipulation is that we can easily avoid numerical overflow by shifting the center of the exponentials, ensuring that the largest value being summed is 1. Numerical underflow may still occur for some exponentials; however the result will still be acceptable.